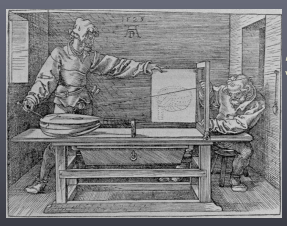
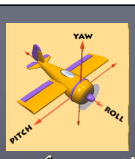




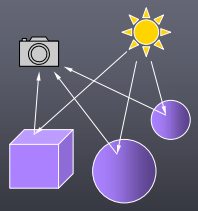
3D-transformationer & Perspektiv

Yngve Sundblad & Gustav Taxén
y@kth.se & gustavt@csc.kth.se

DH2640 Grafik och Interaktionsprogrammering VT 2008

Bakgrund

- Ett smidigt sätt att arbeta med 3D-grafik är att tänka sig att man har en virtuell kamera som ”betraktar” de föremål man vill rita.
- Om vi simulerar hur ljus från ljuskällor reflekteras av ytor och når kameran kan vi få en bild som i bästa fall liknar ett fotografi.



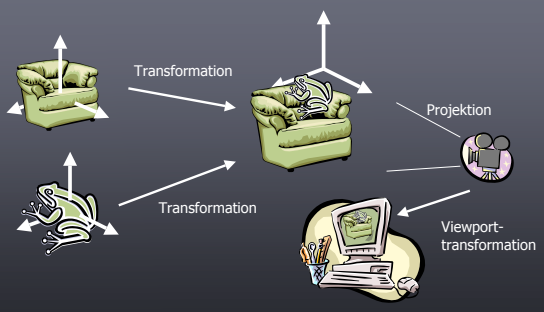
Bakgrund

- Vi börjar med att ta reda på hur vi kan transformera 3D-föremål och gå (projicera) från tre dimensioner till två.
- Vi tar hand om ljusreflektion och skuggning senare i kursen!

Kursmaterial

- 3D-transformationer:
  Kursboken (Angel) kap.4
- Projektioner:
  Kursboken (Angel), kap.5
  Ingrid Carlbom & Joseph Paciorek: Planar Geometric Projections and Viewing Transformations, *Computing Surveys 10*(4), Dec 1978, pp.465-492 (hela -502)
  (i kursbunten)

Transformationer i 3D



Matematiska byggstenar

- Skalär (storhet) u
- Punkt (position) $\mathbf{P}$
- Vektor (längd och riktning) $\mathbf{v}$

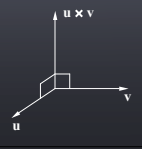
- Kvaternion (3D-rotation)-orientering q

Skalär- och kryssprodukt

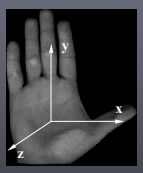
Två av de vanligaste vektoroperationerna i datorgrafik

$\mathbf{u} \cdot \mathbf{v} = (u_x v_x + u_y v_y + u_z v_z) = |\mathbf{u}||\mathbf{v}| \cos\theta$
Skalärprodukten är 0 omm faktorerna är ortogonala (vinkelräta)

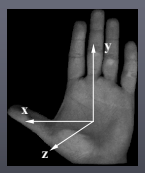
$\mathbf{u} \times \mathbf{v} = (u_y v_z - u_z v_y,\ u_z v_x - u_x v_z,\ u_x v_y - u_y v_x)$
Kryssprodukten (en vektor) är ortogonal mot bägge faktorerna



Höger- och vänsterhänt koordinat-system



OpenGL använder ett **högerhänt** koordinatsystem med **x**-axeln till höger, **y**-axeln uppåt. **z**-axeln pekar utåt ur skärmen.



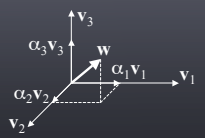
Direct 3D använder ett **vänsterhänt** koordinatsystem med **x**-axeln till höger, **y**-axeln uppåt. **z**-axeln pekar in i skärmen.

I fortsättningen arbetar vi i **högerhänta** koordinatsystem.

Basvektorer

Givet 3 st 3-dimensionella **basvektorer** $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ som alla är vinkelräta mot varandra kan vi skriva vektorn $\mathbf{w}$ som

$$\mathbf{w} = \alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \alpha_3\mathbf{v}_3$$



Basvektorer

Vi inför ”vektorer i vektorer” för att kunna skriva

$$\mathbf{w} = \alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \alpha_3\mathbf{v}_3$$

lite kortare som:

$$\mathbf{w} = \mathbf{a}^T \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \end{pmatrix}$$

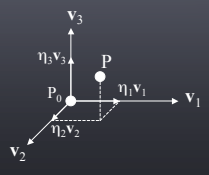
där

$$\mathbf{a} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} \qquad \mathbf{a}^T = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 \end{bmatrix}$$

Ramar (frames)

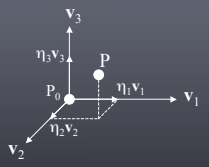
Vi definierar en **ram** som $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathrm{P}_0)$
där P_0 kallas ramens **origo**.
För punkter i denna ram gäller:

$$\mathrm{P} = \mathrm{P}_0 + \eta_1\mathbf{v}_1 + \eta_2\mathbf{v}_2 + \eta_3\mathbf{v}_3$$



Ramar (frames)

eller $\mathrm{P} = \mathbf{p}^T \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathrm{P}_0 \end{pmatrix}$



där

$$\mathbf{p}^T = \begin{bmatrix} \eta_1 & \eta_2 & \eta_3 & 1 \end{bmatrix}$$

(4D homogena koordinater, jfr 3D homogena för punkter i planet)

Vi definierar $1 \cdot \mathrm{P} = \mathrm{P}$ och $0 \cdot \mathrm{P} = 0$

Ramar (frames)

En **riktningsvektor** $\mathbf{w}$ i denna ram skrivs

$$\mathbf{w} = \mathbf{a}^T \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ P_0 \end{pmatrix}$$

där

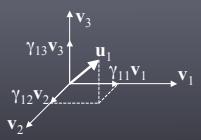
$$\mathbf{a}^T = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & 0 \end{bmatrix}$$

Riktningsvektorer definieras alltså oberoende av origo.

Byte av ram

Givet två ramar $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, P_0)$ och $(\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, Q_0)$, kan vi skriva

$$\begin{aligned} \mathbf{u}_1 &= \gamma_{11}\mathbf{v}_1 + \gamma_{12}\mathbf{v}_2 + \gamma_{13}\mathbf{v}_3 \\ \mathbf{u}_2 &= \gamma_{21}\mathbf{v}_1 + \gamma_{22}\mathbf{v}_2 + \gamma_{23}\mathbf{v}_3 \\ \mathbf{u}_3 &= \gamma_{31}\mathbf{v}_1 + \gamma_{32}\mathbf{v}_2 + \gamma_{33}\mathbf{v}_3 \\ Q_0 &= \gamma_{41}\mathbf{v}_1 + \gamma_{42}\mathbf{v}_2 + \gamma_{43}\mathbf{v}_3 + P_0 \end{aligned}$$



Byte av ram

$$\begin{aligned} \mathbf{u}_1 &= \gamma_{11}\mathbf{v}_1 + \gamma_{12}\mathbf{v}_2 + \gamma_{13}\mathbf{v}_3 \\ \mathbf{u}_2 &= \gamma_{21}\mathbf{v}_1 + \gamma_{22}\mathbf{v}_2 + \gamma_{23}\mathbf{v}_3 \\ \mathbf{u}_3 &= \gamma_{31}\mathbf{v}_1 + \gamma_{32}\mathbf{v}_2 + \gamma_{33}\mathbf{v}_3 \\ Q_0 &= \gamma_{41}\mathbf{v}_1 + \gamma_{42}\mathbf{v}_2 + \gamma_{43}\mathbf{v}_3 + P_0 \end{aligned}$$

Det är smidigare att skriva rambytet på matrisform:

$$\begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \\ Q_0 \end{pmatrix} = \underbrace{\begin{pmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} & 0 \\ \gamma_{21} & \gamma_{22} & \gamma_{23} & 0 \\ \gamma_{31} & \gamma_{32} & \gamma_{33} & 0 \\ \gamma_{41} & \gamma_{42} & \gamma_{43} & 1 \end{pmatrix}}_{\mathbf{M}} \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ P_0 \end{pmatrix}$$

Punkter och rambyte

Ta nu en punkt P och definiera den i två olika ramar:

$P = \alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \alpha_3\mathbf{v}_3 + P_0$ i den första ramen

$P = \beta_1\mathbf{u}_1 + \beta_2\mathbf{u}_2 + \beta_3\mathbf{u}_3 + Q_0$ i den andra

Hur är α- och β-värdena relaterade?

$$P = \begin{bmatrix} \beta_1 & \beta_2 & \beta_3 & 1 \end{bmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \\ Q_0 \end{pmatrix} = \mathbf{b}^T\mathbf{M} \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ P_0 \end{pmatrix} = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & 1 \end{bmatrix} \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ P_0 \end{pmatrix} = \mathbf{a}^T \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ P_0 \end{pmatrix}$$

(enl. ovan)

(föregående bild)

Punkter och rambyte

$$\mathbf{b}^T\mathbf{M} \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ P_0 \end{pmatrix} = \mathbf{a}^T \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ P_0 \end{pmatrix}$$

$$\Leftrightarrow$$

$$\mathbf{b}^T\mathbf{M} = \mathbf{a}^T$$

$$\Leftrightarrow$$

$$\mathbf{a} = \mathbf{M}^T\mathbf{b}$$

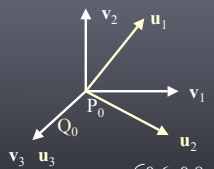
och omvänt:

$$\mathbf{b} = (\mathbf{M}^T)^{-1}\mathbf{a}$$

D.v.s. P's koordinater i ram 2 fås om man multiplicerar koordinaterna för ram 1 med $(\mathbf{M}^T)^{-1}$.

Exempel

Antag att vi har två ramar
$(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, P_0)$ och $(\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, Q_0)$
och att ramarna är relaterade till varandra enligt följande:



$$\begin{aligned} \mathbf{u}_1 &= (3\mathbf{v}_1 + 4\mathbf{v}_2)/5 \\ \mathbf{u}_2 &= (4\mathbf{v}_1 - 3\mathbf{v}_2)/5 \\ \mathbf{u}_3 &= \mathbf{v}_3 \\ Q_0 &= P_0 + 0.5\mathbf{v}_3 \end{aligned}$$

$$\mathbf{M} = \begin{pmatrix} 0.6 & 0.8 & 0 & 0 \\ 0.8 & -0.6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0.5 & 1 \end{pmatrix}$$

$$\mathbf{M}^T = \begin{pmatrix} 0.6 & 0.8 & 0 & 0 \\ 0.8 & -0.6 & 0 & 0 \\ 0 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (\mathbf{M}^T)^{-1} = \begin{pmatrix} 0.6 & 0.8 & 0 & 0 \\ 0.8 & -0.6 & 0 & 0 \\ 0 & 0 & 1 & -0.5 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

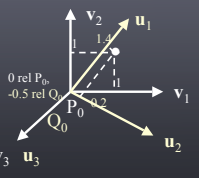
Exemplet

Tag nu punkten $\mathbf{a} = (1\ \ 1\ \ 0\ \ 1)^T$ i den första ramen.
Vilken är dess motsvarighet i den andra ramen?

$$\mathbf{b} = (\mathbf{M}^T)^{-1}\mathbf{a}$$

ger oss

$$\mathbf{b} = \begin{pmatrix} 0.6 & 0.8 & 0 & 0 \\ 0.8 & -0.6 & 0 & 0 \\ 0 & 0 & 1 & -0.5 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \mathbf{a} = \begin{bmatrix} 1.4 & 0.2 & -0.5 & 1 \end{bmatrix}^T$$



Vi kan också se rambytet som en **transformation**.
Exemplet:

Antag att "världen" är ram 1
och att vi gör ett temporärt rambyte till ram 2.

Vi specificerar nu punkten $\mathbf{p} = (1.4\ \ 0.2\ \ -0.5\ \ 1)^T$ i ram 2.

I ram 1 blir den $\mathbf{p}_{\text{världen}} = \mathbf{M}^T\mathbf{p} = (1\ 1\ 0\ 1)^T$

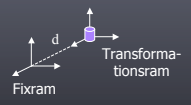
Med andra ord är byte av ram och transformation ekvivalenta begrepp!

Fixram och transformationsram

Ofta jobbar man med en "fixram" (t.ex. kameran/utsiktspunkten) och en "transformationsram".
Fixramen är fast och transformationsramen tillåts variera.
Exempel:

Våra koordinater

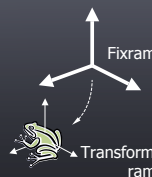
$$\underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -d \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{M}^T} \underbrace{\begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ 1 \end{pmatrix}}_{\text{Punkt i transf.-ramen}} = \underbrace{\begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 - d \\ 1 \end{pmatrix}}_{\text{Motsv. punkt i fixramen}}$$



Fixram och transformationsram

Specificera en modell i "sitt eget" koordinatsystem, **objektkoordinater**.
"Placera ut" modellen i "världen" genom att definiera en transformationsram.
Multiplicera objektkoordinaterna med $\mathbf{M}^T$ för att få modellens koordinater i fixramen.

Groda-modell i dess objektkoordinatsystem



3D-transformationsmatriser(jfr 2D)

Translation

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \mathbf{M}^T \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Skalning

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \mathbf{S} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

3D-rotationsmatriser (tre, en i 2D)

Rotation kring z-axeln

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \mathbf{R}_z(\theta) \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotation kring x-axeln

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \mathbf{R}_x(\theta) \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotation kring y-axeln

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \mathbf{R}_y(\theta) \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Multipla transformationer (som 2D)

Vi kan sätta samman transformationer genom att multiplicera matriserna:

$$\underbrace{\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&-d\\0&0&0&1\end{pmatrix}}_{\mathbf{T}}\underbrace{\begin{pmatrix}1&0&0&0\\0&s&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}}_{\mathbf{S}}=\underbrace{\begin{pmatrix}1&0&0&0\\0&s&0&0\\0&0&1&-d\\0&0&0&1\end{pmatrix}}_{\mathbf{TS}}$$

$$\underbrace{\begin{pmatrix}1&0&0&0\\0&s&0&0\\0&0&1&-d\\0&0&0&1\end{pmatrix}}_{\mathbf{TS}}\begin{pmatrix}\alpha_1\\\alpha_2\\\alpha_3\\1\end{pmatrix}=\begin{pmatrix}\alpha_1\\s\alpha_2\\\alpha_3-d\\1\end{pmatrix}$$

Multipla transformationer

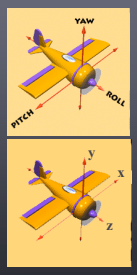
Observera att ordningen på multiplikationen kan spela roll!

$$\underbrace{\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&-d\\0&0&0&1\end{pmatrix}}_{\mathbf{T}}\underbrace{\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&s&0\\0&0&0&1\end{pmatrix}}_{\mathbf{S}}=\underbrace{\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&s&-d\\0&0&0&1\end{pmatrix}}_{\mathbf{TS}}$$

$$\underbrace{\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&s&0\\0&0&0&1\end{pmatrix}}_{\mathbf{S}}\underbrace{\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&-d\\0&0&0&1\end{pmatrix}}_{\mathbf{T}}=\underbrace{\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&s&-sd\\0&0&0&1\end{pmatrix}}_{\mathbf{ST}}$$

Att specificera orientering (rotationer)

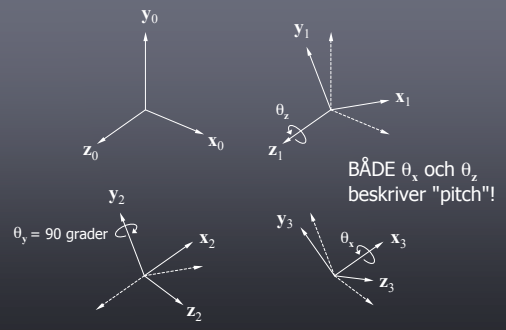
- Vi börjar med att pröva yaw-pitch-roll (gir-stig-roll).
- Antag att flygplanet är modellerat så det initialt "pekar" längs z-axeln.
- Vi roterar först kring z-axeln (roll), sedan kring y-axeln (yaw), sist kring x-axeln (pitch).
- Kallas för att arbeta med **Euler angles** eller **Eulermatriser**.



Gimbal lock

- Om vi roterar en av axlarna så den sammanfaller med en av de andra kan vi hamna i ett läge där vi tappar information!
- Situationen kallas för **Gimbal lock**.
- Beror på att rotationerna alltid görs i samma ordning.

Gimbal lock: exempel



Rotation i 3D

Man kan visa att vilken orientering (3D-rotation) som helst kan beskrivas som en (1 st) rotation kring en vektor.

Rotationsmatriserna (kring x-, y- och z-axlarna) kan sättas ihop till en transformation utgående från rotationsvektorn och rotationsvinkeln

Allmän 3D-rotation - Angel 4.9.4

Om rotationsenhetsvektorn är
$\mathbf{a} = (ax, ay, az)$
och rotationsvinkeln runt denna vektor är
θ
roterar man först kring x-axeln till y-planet (x=0) med vinkeln
$\theta_x = \arctan(ay/az)$; $d=\sqrt{ay^2+az^2}$, $\cos(\theta_x)=az/d$, $\sin(\theta_x)=ay/d$,
sedan kring y-axeln till z-planet med vinkeln
$\theta_y = -\arcsin(ax)$; $\cos(\theta_y)=d$, $\sin(\theta_y)=-ax$,
sedan kring z-axeln med vinkeln $\theta_z = \theta$
sedan tillbaka kring y-axeln med vinkeln $-\theta_y$
sedan tillbaka kring x-axeln med vinkeln $-\theta_x$
Alltså blir totala transformationsmatrisen:
$R_x(-\theta_x)\, R_y(-\theta_y)\, R_z(\theta)\, R_y(\theta_y)\, R_x(\theta_x)$

Kvaternioner, Angel 4.12

En matematisk struktur som kan beskriva 3D-rotation kring en godtycklig vektor är **kvaternioner** (**quaternions**), uppfunna 1843 av irländske matematikern W.R. Hamilton.

Detta motsvarar komplexa talens (Euler mfl, 1700-talet), $i^2=-1$, användning för att beskriva 2D-rotationer i planet:
$z = x + iy = r(\cos\mu + i\sin\mu)$
Rotation vinkeln θ genom multiplikation med
$p = \cos\theta + i\sin\theta$
$p*z = r(\cos(\mu+\theta) + i\sin(\mu+\theta))$

Kvaternioner (quaternions)

En kvaternion q definieras så här:

$$q = (q_0, q_1, q_2, q_3) = (q_0, \mathbf{q})$$
$$\mathbf{q} = q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k},$$
där (jfr komplexa tal)
$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1$$
$$\mathbf{ij}=\mathbf{k},\ \mathbf{jk}=\mathbf{i},\ \mathbf{ki}=\mathbf{j}$$
$$\mathbf{ji}=-\mathbf{k},\ \mathbf{kj}=-\mathbf{i},\ \mathbf{ik}=-\mathbf{j}$$

Kvaternioner - algebra

Lite kvaternionalgebra: tag två kvaternioner a och b:

$$a = (q_0, q_1, q_2, q_3) = (q_0, \mathbf{q})$$
$$b = (p_0, p_1, p_2, p_3) = (p_0, \mathbf{p})$$

Addition och multiplikation:

$$a + b = (p_0 + q_0, \mathbf{p} + \mathbf{q})$$
$$ab = (p_0q_0 - \mathbf{p}\cdot\mathbf{q},\ q_0\mathbf{p} + p_0\mathbf{q} + \mathbf{p}\times\mathbf{q})$$

Norm och invers:

$$|a|^2 = q_0^2 + \mathbf{q}\cdot\mathbf{q}$$
$$a^{-1} = (1/|a|^2)(q_0, -\mathbf{q})$$

Kvaternioner - för 3D-rotation

Tag en punkt P i 3D. Skapa kvaternionen

$$p = (0, P) = x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$$

Välj rotationsaxeln $\mathbf{v}$ och normera den.
Låt rotationsvinkeln vara θ.

Skapa nu kvaternionen r enligt
$$r = (\cos(\theta/2), \sin(\theta/2)\mathbf{v})$$
$$|r|^2 = 1$$
$$r^{-1} = (\cos(\theta/2), -\sin(\theta/2)\mathbf{v})$$

Kvaternioner - för 3D-rotation

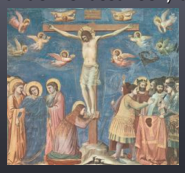
Genom lite räknande visar man att
$p' = rpr^{-1}$
ger samma resultat som sammansatta rotationsmatriserna för att rotera punkten P
θ radianer kring vektorn $\mathbf{v}$.
Om a och b är kvaternioner som representerar rotation enl. ovan gäller också (analogt med komplexa talen för 2D-rotationer) att
$q = ab$
är resultatet av att kombinera rotationerna (först a, sedan b).

Kvaternioner

- Det går alltså att konvertera en kvaternion till en rotationsmatris (och tvärtom)
- Det kan löna sig att lagra orienteringar som kvaternioner: 4 flyttal istället för 3x3 = 9 (för en rotationsmatris)
- Kvaternioner gör det enkelt att interpolera mellan orienteringar (för animering)!

Projektioner - historia

Projektionsritningar:
Arkitekten Vitruvius i Rom, år 14 fKr
Perspektiv:
känt av greker, formaliserat först (centralperspektiv) under renässansen, av arkitekter och målare:

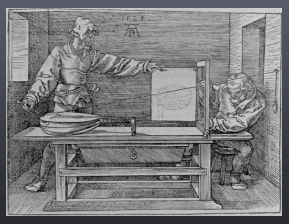


Giotto (1276-1336)

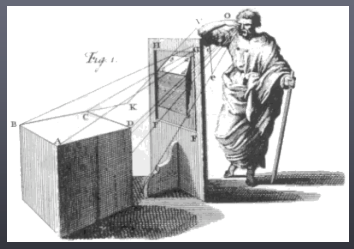
Rafael (1483-1520)

Projektioner - historia

Brunelleschi (1377-1446): Duomo di Firenze mm,
Leonardo da Vinci (1452-1519): Nattvarden mm,
Dürer (1471-1528):

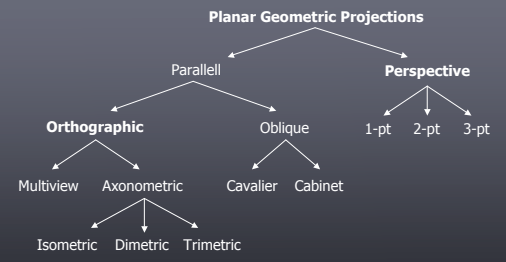


Projektioner - historia



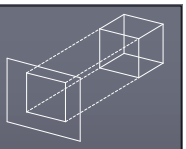
Brook Taylor, **New Principles of Linear Perspective**, 1719.

Projektioner



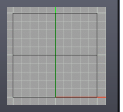
Alla projektionerna finns beskrivna med illustrativa figurer i Carlbom-Pacioreks artikel och i Angel 5.1.

Ortografisk projektion



Projektorerna är vinkelräta mot projektionsytan (på ”oändligt avstånd”).

I enklaste formen är det en eller flera vyer (multiview) med projektionsplan parallella till objektets huvudytor.
Avstånd och vinklar bevaras i projektionerna.
Typiskt exempel: maskinritningar

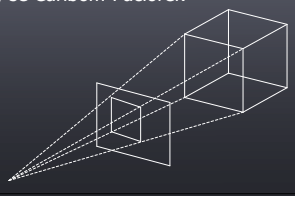


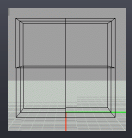
Axonometrisk ortografisk:
Fortfarande vinkelräta projektorer men objektet vridet så att man ser flera ytor.
projektionsplan symmetriskt till alla tre huvudytorna:isometrisk
projektionsplan symmetriskt till två av huvudytorna: dimetrisk
projektionsplan utan sådan symmetri: trimetrisk
Parallellitet bevaras, inte vinklar & avstånd (samma förkortning)

Perspektivprojektion

Projektor på ändligt avstånd.
Storlek minskar med avstånd från projektionsytan.
Avstånd och vinklar bevaras inte i projektionen.
Parallella linjer konvergerar i ”fjärrpunkt” (vanishing point)

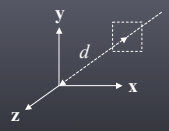
Här enpunkts (ett projektionsplan), finns också två- och tre-, se Carlbom-Paciorek

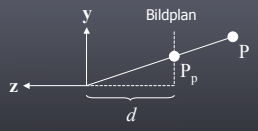




Perspektivprojektion, Angel 5.4

Antag att vi projicerar punkten P på ett bildplan som ligger d enheter från origo och är parallellt med **x**-**y**-planet:



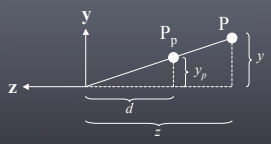


Perspektivprojektion

Likformiga trianglar:

$$P = \begin{bmatrix} x & y & z & 1 \end{bmatrix}^T$$

$$P_p = \begin{bmatrix} x_p & y_p & d & 1 \end{bmatrix}^T$$



$$\frac{y_p}{d} = \frac{y}{z} \Leftrightarrow y_p = \frac{yd}{z} = \frac{y}{(z/d)}$$

Analogt för x_p.
Så den projicerade punkten blir:

$$P_p = \begin{bmatrix} x/(z/d) & y/(z/d) & d & 1 \end{bmatrix}^T$$

Perspektivprojektion

Vi vill kunna skriva perspektivprojektionen som en matris **M**.
För att kunna göra det måste vi införa en ”normeringsoperation”

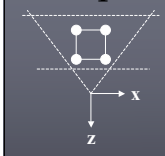
Om resultatet av en transformation blir en punkt

$$P = \begin{bmatrix} x & y & z & w \end{bmatrix}^T$$

med w skilt från 1 ser vi till att alltid dividera med w innan vi fortsätter, d.v.s.

$$P = \begin{bmatrix} x/w & y/w & z/w & 1 \end{bmatrix}^T$$

Perspektivprojektionsmatris

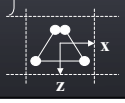


$$\mathbf{M}_P\mathbf{p} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ z/d \end{pmatrix}$$

Eftersom $z/d \neq 1$ dividerar vi alla komponenterna med z/d innan vi använder punkten så att vi får

$$\begin{pmatrix} x/(z/d) \\ y/(z/d) \\ d \\ 1 \end{pmatrix}$$

Detta kallas **perspektivdivision** i OpenGL.



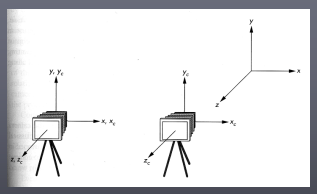
Ortografiskprojektionsmatris

Här är $y_p=y$
och $x_p=x$
och $z_p=-d$
så vi får

$$\mathbf{M}_O\mathbf{p} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -d \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ -d \\ 1 \end{pmatrix}$$

(i boken är d satt till 0)

Att positionera kameran

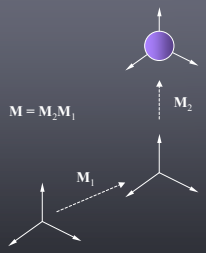


Antingen tänker man att kameran flyttas i förhållande världens ram, **eller** att världsramen flyttas så att den hamnar framför kameran. Operationerna är inverser av varandra.

I OpenGL är kameraramen fix så rent matematiskt gäller det senare alternativet. Men det finns en funktion som hjälper en att "flytta världen" så att det motsvarar att flytta kameran.

Kombinationer av transformationer

- 3D alltså på motsvarande sätt som i 2D.
- Varje transformation motsvarar ett byte av ram.
- Så successiva transformationer är också successiva byten av koordinatsystem!



Inlämningsuppgift 3

Finns på webben senast fredag morgon 29 mars.
Individuell, skickas till y@kth.se senast 11 april.
(Något utsträckt tid)

Individuell inlämningsuppgift 3 till GrIP-kursen vt 2008
Skicka denna blankett ifylld senast 11 april kl. 23.59 till y@kth.se

Ditt namn:

Personnummer:

Uppgiften bedöms på skalan 1-3 (bäst).
Deluppgifterna ger poäng, maximalt 30.
För 3 krävs 25 poäng, för 2 krävs 20 poäng, för 1 krävs 15 poäng.

3D: transformationer, projektioner, kurvor, ytor

(2p) 1. Bevisa att kryssprodukten av två vektorer utom i ett urartat fall (vilket), är ortogonal (vinkelrät) mot båda dessa vektorer.

Fiktiv extenta

Finns på webben senast fredag 4 april.
Genomgås sista föreläsningen, 14 maj.
Tenta 27 maj 8-13, Q-salar

Nästa föreläsning

Fredag 29 mars kl.8.30-10, sal D3 (ej kl.10-12!)